Best Available Copy

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平6－214828

(43)公開日　平成６年(1994)８月５日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/28	３１５　Ａ	9290－5B		

審査請求　有　　請求項の数２　ＯＬ　（全　８　頁）

(21)出願番号　特願平5－5460

(22)出願日　平成５年(1993)１月18日

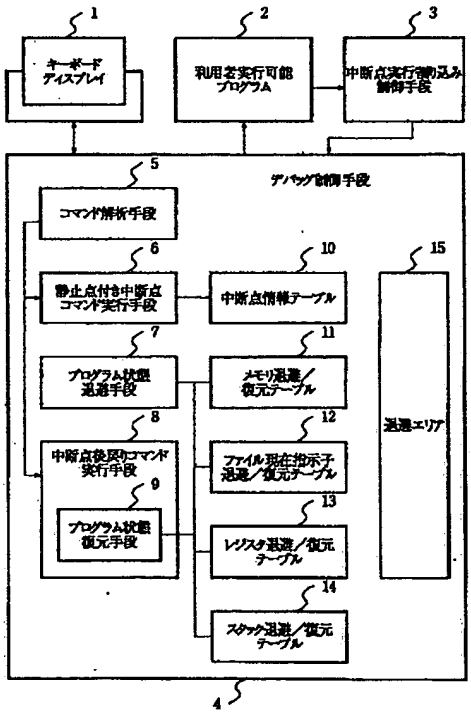
(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　伊藤　邦夫
東京都港区芝五丁目７番１号日本電気株式会社内

(74)代理人　弁理士　京本　直樹　（外２名）

(54)【発明の名称】　対話型デバッグ制御装置

(57)【要約】
【目的】プログラム不正の原因を速やかに発見する。
【構成】デバッグ中に利用者によりプログラム中に中断点を設定する場合に、指定した中断点を後戻り可能な中断点として設定するための静止点付き中断点コマンドを実する静止点付き中断点コマンド実行手段６と、静止点付き中断点コマンドが実行されたことを記憶するための中断点情報テーブル１０と、静止点付き中断点でプログラムが中断されたときに、プログラム状態を退避するためのプログラム状態退避手段７と、利用者により中断点後戻りコマンドが入力されたとき静止点付き中断点コマンドで指定した中断点のプログロム状態に復元するためのプログラム状態復元手段９を含み、静止点付き中断点コマンドで指定した中断点にプログラムを終了させることなく制御を後戻りさせるための中断点後戻りコマンドを実行する中断点後戻りコマンド実行手段８とを含むことを特徴とする対話型デバッグ制御装置。



Best Available Copy

1

【特許請求の範囲】

【請求項1】　利用者実行可能プログラムをキーボードディスプレイ端末等を用いて対話的にデバッグする場合において、

デバッグ中に利用者によりプログラム中に中断点を設定する場合に、指定した中断点を後戻り可能な中断点として設定するための静止点付き中断点コマンドを実行する静止点付き中断点コマンド実行手段と、

前記静止点付き中断点コマンドが実行されたことを記憶するための中断点情報テーブルと、

前記静止点付き中断点でプログラムが中断されたときに、プロラム状態を退避するためのプログラム状態退避手段と、

利用者により前記中断点後戻りコマンドが入力されたとき前記静止点付き中断点コマンドで指定した中断点のプログラム状態に復元するためのプログラム状態復元手段を含み、前記静止点付き中断点コマンドで指定した中断点にプログラムを終了させることなく制御を後戻りさせるための中断点後戻りコマンドを実行する中断点後戻りコマンド実行手段と、

前記プログラム状態退避手段およびプログラム状態復元手段が使用するメモリ退避／復元テーブル、ファイル現在指示子退避／復元テーブル、レジスタ退避／復元テーブルおよびスタック退避／復元テーブルと、

前記利用者実行可能プログラムのプログラム状態を退避する退避エリアとを有することを特徴とする対話型デバッグ制御装置。

【請求項2】　前記プログラム状態退避手段と前記プログラム状態復元手段は、前記利用者実行可能プログラムに含まれたファイル制御ブロック，メモリ退避情報テーブルおよび最大スタックサイズ情報をも使用することを特徴とする請求項1記載の対話型デバッグ制御装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は利用者プログラムの対話型デバッグ制御装置に関する。

【0002】

【従来の技術】利用者プログラムを対話型でデバッグする場合において、利用者はデバッグの対象となるプログラムの直接の不正な実行文をシステムから通知されるメッセージを知して知ることができるが、真の原因はその実行文より前に実行した実行文にある場合が多い。

【0003】例えば、A＝B＋Cの実行文でプログラム例外が発生し、データB，Cの内容を確認したところ、Cに不正データが格納されていたとする。この場合Cに不正データをどこで格納したかを調べる必要があるが、プログラムを再実行し、Cにデータを格納するプログラムポイントすべてに中断点を設定し、少しずつ実行し、不正データがセットされた場所を探していく必要がある。

2

【0004】また、Cに不正データをセットしているのはCを直接参照している実行文であるとは限らない。このため、真の原因となった実行文を通過してしまうと、プログラムを後戻りすることができないので、プログラムを最初から再実行し中断点を再度設定し直さなければならなかった。

【0005】

【発明が解決しようとする課題】上述した従来の対話型デバッグ制御方式では、利用者はプログラムの不正個所を発見するために、プログラムに何個所も中断点を制定しなければならず、また設定した中断点が不正個所より後に実行される個所であった場合にはプログラムを最初から起動しなおして、中断点の範囲にせばめて、再度同様にプログラムを少しずつ実行しなければならなかった。このように、一度のプログラムの実行で不正個所を発見することが困難なため、何度もプログラムを繰り返し実行しなければならず、デバッグの効率が悪いといった欠点がある。

【0006】

【課題を解決するための手段】本発明の装置は、利用者実行可能プログラムをキーボードディスプレイ端末等を用いて対話的にデバッグする場合において、デバッグ中に利用者によりプログラム中に中断点を設定する場合に、指定した中断点を後戻り可能な中断点として設定するための静止点付き中断点コマンドを実行する静止点付き中断点コマンド実行手段と、前記静止点付き中断点コマンドが実行されたことを記憶するための中断点情報テーブルと、前記静止点付き中断点でプログラムが中断されたときに、プロラム状態を退避するためのプログラム状態退避手段と、利用者により前記中断点後戻りコマンドが入力されたとき前記静止点付き中断点コマンドで指定した中断点のプログラム状態に復元するためのプログラム状態復元手段を含み、前記静止点付き中断点コマンドで指定した中断点にプログラムを終了させることなく制御を後戻りさせるための中断点後戻りコマンドを実行する中断点後戻りコマンド実行手段と、前記プログラム状態退避手段およびプログラム状態復元手段が使用するメモリ退避／復元テーブル、ファイル現在指示子退避／復元テーブル、レジスタ退避／復元テーブルおよびスタック退避／復元テーブルと、前記利用者実行可能プログラムのプログラム状態を退避する退避エリアとを有することを特徴とする対話型デバッグ制御装置。

【0007】

【実施例】次に本発明について図面を参照して詳細に説明する。

【0008】本発明の一実施例の構成図である図1を参照すると、本実施例は、利用者が対話型デバッグを行う為のキーボードディスプレイ1と、利用者実行可能プログラム2と、中断点実行割り込み制御手段3と、デバッグ制御手段4とから構成される。再に、デバッグ制御手

Best Available Copy

段4は、コマンド解析手段5と、静止点付き中断点コマンド実行手段6と、プログラム状態退避手段7と、中断点後戻りコマンド実行手段8と、中断点情報テーブル10と、メモリ退避／復元テーブル11と、ファイル現在指示子退避／復元テーブル12と、レジスタ退避／復元テーブル13と、スタック退避／復元テーブル14と、退避エリア15とから構成される。なお、中断点後戻りコマンド実行手段8はプログラム状態復元手断9を含んでいる。

【0009】図2は利用者実行可能プログラム2が生成されるプロセスを示している。先ず、利用者ソースプログラム16が、コンパイラ17で翻訳され、コンパイルユニット18が生成される。コンパイルユニット18の中には、コンパイラ17によって、デバッグ情報テーブル19、ファイル制御ブロック20、メモリ退避情報テーブル21が生成されている。

【0010】デバッグ情報テーブル19は、利用者ソースプログラム内16の中の実行文の行番号とその行番号に対応した実行アドレス、および利用者ソースプログラム16の中で定義したデータ名と属性およびそのデータ名に対応したアドレスから構成されている。ファイル制御ブロック20は、利用者ソースプログラム内16の中で定義したファイルに関する属性および制御情報から構成されている。また、メモリ退避情報テーブル21は利用者ソースプログラム内16の中の利用者データ領域で静止点時に退避される必要のある領域のアドレスと長さから構成されている。

【0011】リンカ22は、コンパイルユニット18を入力し、利用者実行可能プログラム2を生成する。このとき、コンパイルユニット18の外に最大スタックサイズ情報23を作成する。

【0012】図3は利用者プログラム2をデバッグする場合の具体例について説明するためのキーボードディスプレイデバッグ画面と利用者ソースプログラム例を示している。この利用者プログラム16では行番号1260の実行文でデータ名Cの内容が不正のためプログラム例外が発生するが、これは行番号680行の実行文でデータ名Yの領域の後ろ2バイト（データ名Xの領域と重なる）を破壊したことが真の原因である。キーボードディスプレイデバッグ画面には、利用者ソースプログラム16の不正個所を発見するための実際のデバッグコマンドの入力手順を示している。

【0013】図4はデバッグ制御手段が作成する各種のテーブルの構造を示している。

【0014】中断点情報テーブル10は、中断点アドレスとその中断点が静止点付きであるか否かを示す静止点SWから構成されている。

【0015】メモリ退避／復元テーブル11は利用者実行可能プログラム2の退避すべきメモリの退避元アドレスと、退避先エリアのアドレスと、退避メモリの長さと

から構成されている。デバッグ制御手段4は、退避元アドレスと長さを、利用者実行可能プログラム2の中に生成されているメモリ退避情報テーブル21から取り込み、退避先アドレスは動的に生成した退避エリア15のアドレスを格納する。

【0016】ファイル現在指示子退避／復元テーブル12はファイル制御ブロックのアドレス、退避先のアドレス、ファイル制御ブロックの長さから構成されている。ファイル制御ブロックのアドレスと、ファイル制御ブロックの長さは、利用者実行可能プログラム2の中に生成されているファイル制御ブロック20から取り込み、退避先アドレスは動的に生成した退避エリア15のアドレスを格納する。

【0017】レジスタ退避／復元テーブル13は退避先アドレスのみから構成される。このレジスタの退避エリアの長さはシステムにより固定のため不要である。

【0018】スタック退避／復元テーブル14は、スタック領域のアドレスと、退避先アドレスとスタック領域の長さから構成される。スタック領域のアドレスは、利用者実行可能プログラム2のスタックをたどることで得られ、スタック領域の長さは、利用者実行可能プログラム2の中の最大スタックサイズ23から取り込む。図5はデバッグ制御手段4の処理フローである。

【0019】以下に、図3，図4及び図5をも参照して本実施例の動作について説明する。

【0020】先ず、利用者実行可能プログラム2がデバッグモードで起動されると、オペレーティングシステム（図示せず）は、利用者実行可能プログラム2を実行記憶域にロードした後、デバッグ制御手段4を呼び出し、図5のA1の制御が渡される。デバッグ制御手段4は、デバッグ開始メッセージをキーボードディスプレイ1に表示（図5のA2，図3の（1））した後、利用者からのデバッグコマンドを待つ（図5のA3）。

【0021】利用者が静止点付き中断点コマンド（図3の（2））を入力するとコマンド解析手段5はそのコマンドの解析を行い（図5のA4）、静止点付き中断点コマンド実行手段6に制御を移す（図5のA5）。静止点付き中断点コマンドの実行では、利用者ソースプログラム16の300行目に対応した利用者実行可能プログラム2中の中断点アドレスをデバッグ情報テーブル19を参照して取り出し、中断点情報テーブル10の中断点アドレスにセットする。また、利用者ソースプログラム16の行番号300に対応した実行記憶域上の命令コードが実行されると、例外割り込みが発生して中断点実行割り込み制御手段3に制御が渡さるようにしておくとともに、静止点付き中断点であることを示すため中断点情報テーブル10の静止点SWをオンにしておく。以上で静止点付き中断点コマンドの実行（図5のA5）の処理を終わり、利用者からの次のデバッグコマンドを待つ（図5のA3）。

Best Available Copy

5

【００２２】キーボードディスプレイ１から次に利用者がＧＯコマンドを入力（図３の（３））をすると、デバッグコマンド解析手段４は前記と同様にコマンドの解析を行い、ＧＯコマンドの実行（図５のＡ１０）に制御を移す。ＧＯコマンドの実行では、デバッグ制御手段４の処理を一旦終了し、オペレーティングシステムに制御を戻すオペレーティングシステムは利用者実行可能プログラム２に制御を渡し利用者実行可能プログラム２の先頭から実行が始まる。

【００２３】利用者実行可能プログラム２の実行でソースプログラム１６の３００行目に対応した命令を実行すると、前記静止点き中断点コマンド実行手段６の実行（図５のＡ５）により例外割り込みが発生し、中断点実行割り込み制御手段３に制御がわたる。中断点実行割り込み制御手段３は、デバッグ制御手段２の二次入り口（図５のＡ１１）を呼び出す。デバッグ制御手段２は中断発生メッセージ（図３の（４））を出力（図５のＡ１２）する。その後、プログラム状態退避手段７を実行する（図５のＡ１３）。

【００２４】プログラム状態退避手段７は、まずメモリ退避／復元テーブル１１が作成されているか調べ、作成されていなければ作成する。メモリ退避／復元テーブル１１の退避元のアドレスは、利用者実行可能プログラム２の中のメモリ退避情報テーブル２１を参照してセットする。メモリ退避／復元テーブル１１の退避先のアドレスは、利用者実行可能プログラム２の中のメモリ退避情報テーブル２１を参照して退避メモリの長さを取り出し、その長さと同じ大きさの退避エリア１５を動的に生成しその先頭アドレスをセットする。メモリ退避／復元テーブル１１の長さにはメモリ退避情報テーブル２１を参照してセットする。この操作は、利用者実行可能プログラム２の中のメモリ退避情報テーブル２１のすべてのアイテムについて繰り返される。

【００２５】続いてプログアム状態退避手段７は、ファイル現在指示子退避／復元テーブル１１が作成されているか調べ、作成されていなければ作成する。ファイル現在指示子退避／復元テーブル１２のファイル制御ブロックのアドレスには、利用者実行可能プログラム２の中のファイル制御ブロックのアドレスをセットする。ファイル現在指示子退避／復元テーブル１２の退避先のアドレスには、ファイル制御ブロックに含まれるファイルの種類を調べその種類毎に決められた長さと同じ大きさの退避エリア１５を動的に生成し、その先頭アドレスをセットする。ファイル現在指示子退避／復元テーブル１２の長さには退避されるファイル制御ブロックの大きさをセットする。この操作は、利用者実行可能プログラム２の中のすべてのファイル制御ブロックについて繰り返される。

【００２６】続いてプログラム状態退避手段７は、レジスタ退避／復元テーブル１３が作成されているか調べ、

6

作成されていなければ作成する。レジスタの数と大きさはハードウェアにより決められているため、そのすべてのレジスタを退避するに充分な大きさの退避エリア１５を動的に生成しその先頭アドレスをセットする。

【００２７】続いてプログラム状態退避手段７は、スタック退避／復元テーブル１４が作成されているか調べ、作成されていなければ作成する。スタック退避／復元テーブル１４のスタック領域のアドレスは、利用者実行可能プログラムの実行中のスタックをたどることによって得られ、その先頭アドレスがセットされる。長さには利用者実行可能プログラム２の中の最大スタックサイズから取り込む。スタック退避／復元テーブル１４の退避先アドレスは最大スタックの長さと同じ大きさの退避エリア１５を動的に生成しその先頭アドレスをセットする。

【００２８】以上のように、プログラム状態退避手段７は、メモリ退避／復元テーブル１１、ファイル現在指示子退避／復元テーブル１２、レジスタ退避／復元テーブル１３、スタック退避／復元テーブル１４を作成すると、ソースプログラム１６の行番号３００の実行時点の利用者プログラムのユーザメモリと、ファイルの現在指示子と、レジスタと、スタックを退避エリア１５に前記テーブルを使用して退避し処理を完了する。

【００２９】デバッグ制御手段２は次のデバッグコマンドの入力を待つ（図５のＡ３）。利用者は次にソースプログラム１６の行番号３５００に中断点を設定するためのコマンド（図３の（５））を入力する。デバッグ制御手段２は、中断点情報テーブルの静止点ＳＷをオフにすることを除いて、前記静止点付き中断点コマンドの実行（図５のＡ５）と同じ処理をソースプログラム１６の行番号３５００に対応する利用者実行可能プログラム２の中の命令アドレスに対して行う。その後、次のデバッグコマンドの入力を待つ（図５のＡ３）。

【００３０】利用者がデバッグ画面からＧＯコマンドを入力（図３の（６））すると、デバッグ制御手段２は、制御を利用者実行可能プログラム２に戻す。利用者実行可能プログラム２はソースプログラム１６の行番号３００に対応する命令から実行を再開する。利用者実行可能プログラム２の実行中にソースプログラム１６の行番号１２６０に対応する命令でプログラム例外が発生する。

【００３１】この結果、制御は利用者実行可能プログラム２から二次入り口（図５のＡ１１）を呼び出す。デバッグ制御手段は中断発生メッセージ（図５のＡ１２）をキーボードディスプレイ１に出力（図３の（８））に出力し、更にプログラム例外が発生した命令に対応するソースプログラム上の行番号をデバッグ情報テーブル１９から取り出し、キーボードディスプレイ１に出力（図３の（９））に出力する。その後、利用者からのデバッグコマンドを待つ（図５のＡ３）。

【００３２】利用者はソースプログラムの行番号１２６０の文でプログラム例外が発生したこと知り、その文

Best Available Copy

（A＝B＋C）のBのデータの内容を調べるため、DUMPコマンドを入力する（図3の（10））。デバッグ制御手段2はコマンド解析手段5を実行し（図5のA4）、DUMPコマンドを識別したのちDUMPコマンドの実行（図5のA6）を行う。DUMPコマンドの実行（図5のA6）は、利用者実行可能プログラム2に含まれるデバッグ情報テーブル19を参照し、利用者データBのアドレスを取り込み、そのアドレスが指す内容をキーボードディスプレイ1に表示する（図3の（11））。

【0033】利用者はデータBの内容が正常であることを確認した後、次の利用者データCの内容を調べるため、DUMPコマンドを入力する（図3の（12））。デバッグ制御手段2はコマンド解析手段5を実行（図5のA4）し、DUMPコマンドを識別したのちDUMPコマンドの実行（図5のA6）を行う。DUMPコマンドの実行（図5のA6）では、利用者実行可能プログラム2に含まれるデバッグ情報テーブル19を参照し、利用者データCのアドレスを取り込み、そのアドレスが指す内容をキーボードディスプレイ1に表示する（図3の（13））。これにより、利用者はデータCの内容が不正であることを知る。

【0034】このため、直前にCにデータを代入した命令（ソースプログラム16の行番号890）に中断点コマンド（図3の（14））を入力する。デバッグ制御手段2は、中断点情報テーブルの静止点SWをオフにすることを除いて、前記静止点付き中断点コマンドの実行（図5のA5）と同じ処理をソースプログラム16の行番号890に対応する利用者実行可能プログラム2の中の命令アドレスに対して行う。その後、次のデバッグコマンドの入力を待つ（図5のA3）。

【0035】利用者はデバッグ画面から利用者実行可能プログラム2を直前の静止点の状態を戻す。このため、中断点後戻りコマンド（図3の（15））を入力する。デバッグ制御手段2はコマンド解析手段5を実行（図5のA4）し、中断点後戻りコマンドを識別した後、中断点後戻りコマンド実行手段8を実行する（図5のA8）。中断点後戻りコマンド実行手段8は、以前に静止点付き中断点コマンドが実行されているか中断点情報テーブル10を調べる。既に静止点付き中断点コマンドが実行されていれば、プログラム状態復元手段の実行（図5のA9）を行う。

【0036】プログラム状態復元手段の実行（図5のA9）は、以前にプログラム状態退避手段7によって作成された、メモリ退避／復元テーブル11、ファイル現在指示子退避／復元テーブル12、レジスタ退避／復元テーブル13、スタック退避／復元テーブル14を使って、退避エリア15に保存されているソースプログラム16の行番号300の実行時点の利用者プログラムのユーザメモリと、ファイルの現在指示子と、レジスタと、

スタックの内容を利用者実行可能プログラム2上の対応した各領域に戻す。

【0037】その後、デバッグ制御手段2は、制御をソースプログラム16の行番号300に対応する利用者実行可能プログラム2の命令から実行を再開する。利用者実行可能プログラム2は、ソースプログラム16の行番号890に対応する利用者実行可能プログラム2の命令を実行すると、前記中断点コマンドの実行（図3の（14））により例外割り込みが発生し、中断点実行割り込み制御手段3に制御がわたる。

【0038】中断点実行割り込み制御手段3は、デバッグ制御手段2の二次入り口（図5のA1）を呼び出す。デバッグ制御手段2は中断メッセージ（図3の（16））を出力（図5のA12）する。デバッグ制御手段2は中断点割り込みが発生したアドレスで中断点情報テーブル10を検索し、静止点付きの中断点でないことを判定すると、次のデバッグコマンドの入力を待つ（（図5のA3）。利用者はソースプログラム16の行番号890の命令から利用者データXの内容をみるため、DUMPコマンドを入力（図3の（17））する。

【0039】デバッグ制御手段2はコマンド解析手段を実行（図5のA4）し、DUMPコマンドを識別したのちDUMPコマンドの実行（図5のA6）を行う。DUMPコマンドの実行（図5のA6）は、利用者実行可能プログラム2に含まれるデバッグ情報テーブル19を参照し、利用者データXのアドレスを取り込み、そのアドレスが指す内容をキーボードディスプレイ1に表示する（図3の（18））。利用者はデータXの内容が不正であることを知る。

【0040】データXはソースプログラム上で行番号310で代入されている以外に値の設定を行っていないことから、データXの領域が破壊される可能性のある命令をソースプログラム16から探す。ソースプログラム16の行番号680の命令が可能性があることを判断し、Nの内容を参照するためDUMPコマンドを入力する（図3の（19））。デバッグ制御手段2はコマンド解析手段5を実行（図5のA4）し、DUMPコマンドを識別したのちDUMPコマンドの実行（図5のA6）を行う。

【0041】DUMPコマンドの実行（図5のA6）は、利用者実行可能プログラム2に含まれるデバッグ情報テーブル19を参照し、利用者データNのアドレスを取り込み、そのアドレスが指す内容をキーボードディスプレイ1に表示する（図3の（18））。利用者はNの内容が12であることを知り行番号680の命令がデータYの領域の後ろの2バイトを壊すことによりに不正な値が代入されることを知る。そしてデータNの値はソースプログラム16の行番号670で設定しており、その命令の誤りであることをしる。以上のようにして、利用者は、何度でもプログラムを好きな時点から再実行させ

Best Available Copy

9

不正個所を究明していくことができる。

【0042】

【発明の効果】本発明によると、プログラムの流れを後戻り可能にする中断点の設定ができるために、一回のプログラムの実行中に何度でも同一のプログラム部分を再実行させることにより、プログラム不正の原因を速やかに発見できるという利点がある。

【図面の簡単な説明】

【図1】本発明の一実施例の全体構成図である。

【図2】利用者実行可能プログラムに含まれる情報の生成過程を示す図である。

【図3】利用者プログラムをキーボードディスプレイ端末で実際にデバッグをした場合の具体例を示す図である。

【図4】デバッグ制御手段が作成する各種のテーブルの構造を示す図である。

【図5】デバッグ制御手段の動作を示す図である。

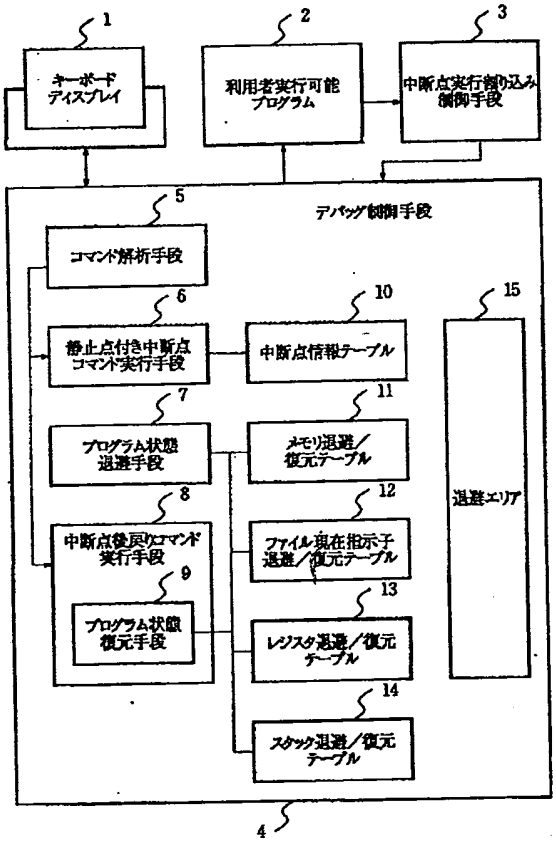
【符号の説明】

1　キーボードディスプレイ
2　利用者実行可能プログラム
3　中断点実行割込み制御手段

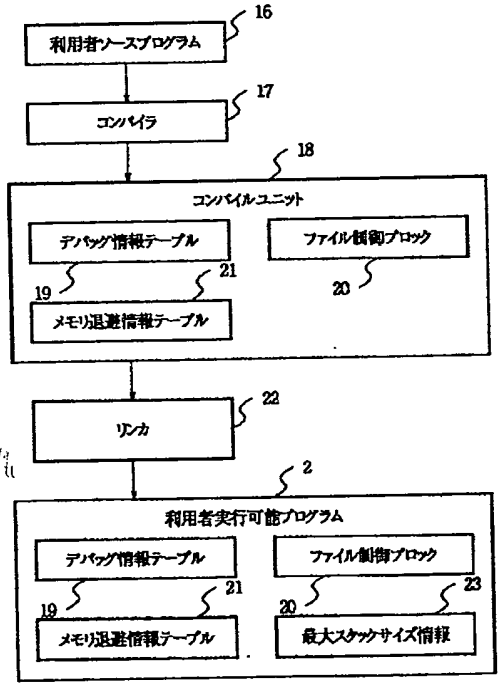
10

4　デバッグ制御手段
5　コマンド解析手段
6　静止点付き中断点コマンド実行手段
7　プログラム状態退避手段
8　中断点後戻りコマンド実行手段
9　プログラム状態復元手段
10　中断点情報テーブル
11　メモリ退避／復元テーブル
12　ファイル復元指示子退避／復元テーブル
13　レジスタ退避／復元テーブル
14　スタック退避／復元テーブル
15　退避エリア
16　利用者ソースプログラム
17　コンパイラ
18　コンパイルユニット
19　デバッグ情報テーブル
20　ファイル制御ブロック
21　メモリ退避情報テーブル
22　リンカ
23　最大スタックサイズ情報

【図1】

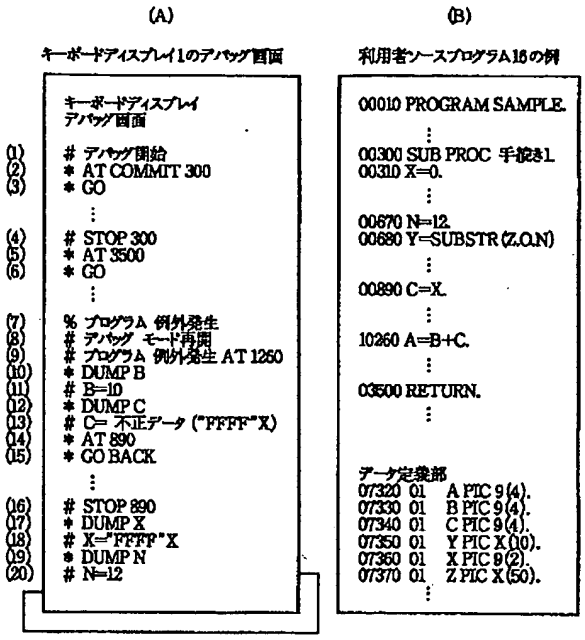


【図2】

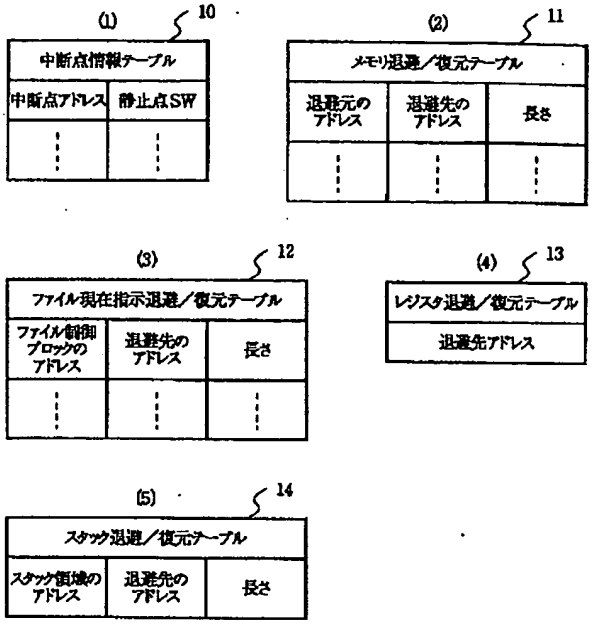


Best Available Copy

【図3】



【図4】



Best Available Copy

【図5】